Attacking the Common Algorithmic Problem by Recognizer P Systems

Mario J. Pérez Jiménez and Francisco J. Romero Campero

Avda. Reina Mercedes s/n, 41012, Sevilla, España
{Mario.Perez, Francisco-Jose.Romero}@cs.us.es

Abstract. Many **NP**-complete problems can be viewed as special cases of the Common Algorithmic Problem (CAP). In a precise sense, which will be defined in the paper, one may say that CAP has a property of *local universality*. In this paper we present an effective solution to the decision version of the CAP using a family of recognizer P systems with active membranes. The analysis of the solution presented here will be done from the point of view of complexity classes in P systems.

Keywords: Membrane Computing, Common Algorithmic Problem, Cellular Complexity Classes.

1 Introduction

Membrane Computing is an emergent branch of Natural Computing. This unconventional model of computation is a kind of distributed parallel system, and it is inspired by some basic features of biological membranes.

Since Gh. Paun introduced it in [4], many different classes of such computing devices, called P systems, have already been investigated. Most of them are *computationally universal*, i.e., able to compute whatever a Turing machine can do, as well as *computationally efficient*, i.e., are able to trade space for time and solve in this way presumably intractable problems in a feasible time.

This paper deals with the Common Algorithmic Problem. This problem has the nice property (we can call this property *local universality*) that several other **NP**–complete problems can be reduced to it in linear time – we can say that they are *subproblems* of CAP. This property was already considered in [2], will be precisely defined in Section 2, and further illustrated in the paper.

Our study is focused on the design of a family of recognizer P systems solving it. We have followed the ideas and schemes used to solve other numerical **NP**-complete problems, such as Subsetsum in [6] and the Knapsack problem in [7].

* This work is supported by the Ministerio de Ciencia y Tecnología of Spain, by the *Plan Nacional de I+D+I (2000–2003)* (TIC2002-04220-C03-01), cofinanced by FEDER funds, and by a FPI fellowship (of the second author) from the University of Seville.

Due to the strong similarities of the design of these solutions the idea of a *cellular programming language* seems possible as it is suggested in [1].

The analysis of the presented solution will be done from the point of view of the complexity classes presented within the framework of the *complexity classes in P systems* studied in [5] and [9].

The paper is organized as follows. In the next section the Common Algorithmic Problem is presented as well as six **NP**-complete problems that can be viewed as "subproblems" of it. Section 3 recalls recognizer P systems with active membranes. In section 4 a polynomial complexity class for P systems is briefly introduced. Sections 5, 6 and 7 present a cellular solution to the Common Algorithmic Decision Problem. Conclusions are given in section 8.

2 The Common Algorithmic Problem

The *Common Algorithmic Problem* (CAP) [2] is the following: *let S be a finite set and F be a family of subsets of S. Find the cardinality of a maximal subset of S which does not include any set belonging to F. The sets in F are called forbidden sets.*

The Common Algorithmic Problem is an optimization problem, which can be transformed into a roughly equivalent decision problem by supplying a target value to the quantity to be optimized, and asking the question whether or not this value can be attained.

The Common Algorithmic Decision Problem (CADP) is the following: *Given S a finite set, F a family of subsets of S, and $k \in \mathbf{N}$, we ask if there exists a subset A of S such that $|A| \geq k$, and which does not include any set belonging to F.*

Definition 1. *We say that a problem X is a subproblem of another problem Y if there exists a linear–time reduction from X to Y (using a logarithmic bounded space).*

That is, X is a subproblem of Y if we can pass from problem X to problem Y through a simple rewriting process.

Next, we present some **NP**–complete problems that are subproblems of the CAP (or CADP).

2.1 The Maximum Independent Set Problem

The *Maximum Independent Set Problem* (MIS) is the following: *Given an undirected graph G, find the cardinality of a maximal independent subset I of G.*

The *Independent Set Decision Problem* (ISD) is the following: *Given an undirected graph G, and $k \in \mathbf{N}$, determine whether or not G has an independent set of size at least k.*

Theorem 1. *MIS (resp. ISD) is a subproblem of CAP (resp. CADP).*

2.2 The Minimum Vertex Cover Problem

The *Minimum Vertex Cover Problem* (MVC) is the following: *Given an undirected graph G, find the cardinality of a minimal set of a vertex cover of G.*

The *Vertex Cover Decision Problem* (VCD) is the following: *Given an undirected graph G, and $k \in \mathbf{N}$, determine whether or not G has a vertex cover of size at most k.*

Theorem 2. *MVC (resp. VCD) is a subproblem of CAP (resp. CADP).*

2.3 The Maximum Clique Problem

The *Maximum Clique Problem* (MAX-CLIQUE) is the following: *Given an undirected graph G, find the cardinality of a largest clique in G.*

The *Clique Decision Problem* is the following: *Given an undirected graph G, and $k \in \mathbf{N}$, determine whether or not G has a clique of size at least k.*

Theorem 3. *MAX-CLIQUE (resp. Clique Decision problem) is a subproblem of CAP (resp. CADP).*

2.4 The Satisfiability Problem

The *Satisfiability Problem* (SAT) is the following: *For a given set U of boolean variables and a finite set C of clauses over U, is there a satisfying truth assignment for C?*

Theorem 4. *Let $\varphi \equiv c_1 \wedge \cdots \wedge c_p$ be a boolean formula in conjunctive normal form. Let $Var(\varphi) = \{x_1, \ldots, x_n\}$, $c_i = l_{i,1} \vee \cdots \vee l_{i,r_i}$ $(1 \leq i \leq p)$, and $S = \{x_1, \ldots, x_n\} \cup \{\overline{x}_1, \ldots, \overline{x}_n\}$. For each i $(1 \leq i \leq p)$ let $A_i = \{\overline{l}_{i,1}, \ldots, \overline{l}_{i,r_i}\}$, considering $\overline{\overline{x}} = x$. Let $F = \{\{x_1, \overline{x}_1\}, \ldots, \{x_n, \overline{x}_n\}, A_1, \ldots, A_p\}$. Then the formula φ is satisfiable if and only if the solution of the CAP on input (S, F) is n.*

2.5 The Undirected Hamiltonian Path Problem

The *Undirected Hamiltonian Path Problem* is the following: *Given an undirected graph and two distinguished nodes u, v, determine whether or not there exists a path from u to v visiting each node exactly once.*

Theorem 5. *Let $G = (V, E)$ be an undirected graph, with $V = \{v_1, \ldots, v_n\}$. Then the following conditions are equivalent:*

(a) The graph G has a Hamiltonian path from v_1 to v_n.
(b) The solution of the CAP on input (S, F) is $n - 1$, where: $S = E$, and $F = \bigcup_{i=1}^{n} F_i$, with $F_i = \{B : B \subseteq B_i \ |B| = 2\}$, for $i = 1, n$, and $F_i = \{B : B \subseteq B_i \wedge |B| = 3\}$, for all $1 < i < n$, with $B_i = \{\{v_i, u\} : \{v_i, u\} \in E\}$.

2.6 The Tripartite Matching Problem

The *Tripartite Matching Problem* is the following: *Given three sets B, G, and H, each containing n elements, and a ternary relation $T \subseteq B \times G \times H$, determine whether or not there exists a subset T' of T such that $|T'| = n$ and no two of triples belonging to T' have a component in common.*

We say that T' is a tripartite matching associated with (B, G, H, T).

Theorem 6. *Let $B = \{b_1, \ldots, b_n\}$, $G = \{g_1, \ldots, g_n\}$, and $H = \{h_1, \ldots, h_n\}$, be sets containing n elements. Let T be a subset of $B \times G \times H$. Then the following conditions are equivalent:*

(a) There exists a tripartite matching associated with (B, G, H, T).
(b) The solution of the CAP on input (S, F) is n, where $S = T$ and $F = \bigcup_{i=1}^{n}(F_{b_i} \cup F_{g_i} \cup F_{h_i})$, with $F_{b_i} = \{A : |A| = 2 \wedge A \subseteq \{(b_i, g, h) : (b_i, g, h) \in T\}\}$, $F_{g_i} = \{A : |A| = 2 \wedge A \subseteq \{(b, g_i, h) : (b, g_i, h) \in T\}\}$, and $F_{h_i} = \{A : |A| = 2 \wedge A \subseteq \{(b, g, h_i) : (b, g, h_i) \in T\}\}$, for all $1 \leq i \leq n$.

3 Recognizer P Systems with Active Membranes

Definition 2. *A* P system with input *is a tuple (Π, Σ, i_Π), where: (a) Π is a P system, with working alphabet Γ, with p membranes labelled by $1, \ldots, p$, and initial multisets $\mathcal{M}_1, \ldots, \mathcal{M}_p$ associated with them; (b) Σ is an (input) alphabet strictly contained in Γ; the initial multisets are over $\Gamma - \Sigma$; and (c) i_Π is the label of a distinguished (input) membrane.*

Let m be a multiset over Σ. The *initial configuration of (Π, Σ, i_Π) with input* m is $(\mu, \mathcal{M}_1, \ldots, \mathcal{M}_{i_\Pi} \cup m, \ldots \mathcal{M}_p)$.

Definition 3. *Let $\mu = (V(\mu), E(\mu))$ be a membrane structure. The* membrane structure with environment *associated with μ is the rooted tree such that: (a) the root of the tree is a new node that we will denote env; (b) the set of nodes is $V(\mu) \cup \{env\}$; and (c) the set of edges is $E(\mu) \cup \{\{env, skin\}\}$. The node env is called the* environment *of the structure μ.*

In the case of P systems with input and with external output, the concept of computation is introduced in a similar way as for standard P systems – see [3]– but with a slight change. Now the configurations consist of a membrane structure with environment, and a family of multisets of objects associated with each region and with the environment.

Next we introduce P systems able to accept or reject multisets considered as inputs. Therefore, these systems will be suitable to attack the solvability of decision problems.

Definition 4. *A recognizer P system is a P system with input (Π, Σ, i_Π), and with external output such that: (a) the working alphabet contains two distinguished elements YES, NO; (b) all its computations halt; and (c) if $\mathcal{C}$ is a computation of Π, then either the object YES or the object NO (but not both) have to be sent out to the environment, and only in the last step of the computation.*

Definition 5. *We say that $\mathcal{C}$ is an accepting computation (respectively, rejecting computation) if the object YES (respectively, NO) appears in the environment associated with the corresponding halting configuration of $\mathcal{C}$.*

These recognizer P systems are specially adequate when we are trying to solve a decision problem. In this paper we will deal with P systems with active membranes. We refer to [3] (see chapter 7, section 7.2) for a detailed definition of evolution rules, transition steps, configurations and computations in this model.

Let us denote by $\mathcal{AM}$ the class of recognizer P systems with active membranes using 2-division (for elementary membranes).

4 The Complexity Class $\mathbf{PMC}_{\mathcal{F}}$

Roughly speaking, a computational complexity study of a solution to a problem is an estimation of the resources (time, space, etc) that are required through all processes that take place in the way from the bare instance of the problem up to the final answer.

The first results about "solvability" of **NP**–complete problems in polynomial time (even linear) by cellular computing systems with membranes were obtained using variants of P systems that lack an input membrane. Thus, the constructive proofs of such results need to design one system for each instance of the problem.

This drawback can be easily avoided if we consider P systems with input. Then, the same system could solve different instances of the problem, provided that the corresponding input multisets are introduced in the input membrane.

Instead of looking for a single system that solves a problem, we prefer designing a family of P systems such that each element of the family decides all the instances of "equivalent size" of the problem.

Let us now introduce some basic concepts before the definition of the complexity class itself.

Definition 6. *Let L be a language, and $\mathbf{\Pi} = (\Pi(n))_{n\in\mathbf{N}}$ be a family of P systems with input. A polynomial encoding of L in $\mathbf{\Pi}$ is a pair (g, h) of polynomial-time computable functions $g : L \to \bigcup_{n\in\mathbf{N}} I_{\Pi(n)}$ and $h : L \to \mathbf{N}$, such that for every $u \in L$ we have $g(u) \in I_{\Pi(h(u))}$.*

That is, for each string $u \in L$, we have a multiset $g(u)$ and a number $h(u)$ associated with it such that $g(u)$ is an input multiset for the P system $\Pi(h(u))$.

Lemma 1. *Let $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$ be languages. Let $\mathbf{\Pi} = (\Pi(n))_{n\in\mathbf{N}}$ a family of P systems with input. If $r : \Sigma_1^* \to \Sigma_2^*$ is a polynomial time reduction from L_1 to L_2, and (g, h) is a polynomial encoding of L_2 in $\mathbf{\Pi}$, then $(g \circ r, h \circ r)$ is a polynomial encoding of L_1 in $\mathbf{\Pi}$.*

For a detailed proof, see [9].

Definition 7. *Let $\mathcal{F}$ be a class of recognizer P systems. A decision problem $X = (I_X, \theta_X)$ is solvable in polynomial time by a family of P systems $\mathbf{\Pi} = (\Pi(n))_{n\in\mathbf{N}}$ from $\mathcal{F}$, and we denote this by $X \in \mathbf{PMC}_{\mathcal{F}}$, if the following is true:*

- *The family* $\mathbf{\Pi}$ *is consistent with regard to the class* $\mathcal{F}$*; that is,* $\forall t \in \mathbf{N}$ $(\Pi(t) \in \mathcal{F})$.
- *The family* $\mathbf{\Pi}$ *is polynomially uniform by Turing machines; that is, there exists a deterministic Turing machine constructing* $\Pi(t)$ *from* t *in polynomial time.*
- *There exists a polynomial encoding* (g, h) *from* I_X *to* $\mathbf{\Pi}$ *such that:*
  - *The family* $\mathbf{\Pi}$ *is polynomially bounded with regard to* (X, g, h)*; that is, there exists a polynomial function* p*, such that for each* $u \in I_X$ *every computation of* $\Pi(h(u))$ *with input* $g(u)$ *is halting and, moreover, it performs at most* $p(|u|)$ *steps.*
  - *The family* $\mathbf{\Pi}$ *is sound with regard to* (X, g, h)*; that is, for each* $u \in I_X$*, if there exists an accepting computation of* $\Pi(h(u))$ *with input* $g(u)$*, then* $\theta_X(u) = 1$.
  - *The family* $\mathbf{\Pi}$ *is complete with regard to* (X, g, h)*; that is, for each* $u \in I_X$*, if* $\theta_X(u) = 1$*, then every computation of* $\Pi(h(u))$ *with input* $g(u)$ *is an accepting one.*

In the above definition we have imposed to every P system $\Pi(n)$ to be *confluent*, in the following sense: every computation with the *same* input produces the *same* output; that is, for every input multiset m, either every computation of $\Pi(n)$ with input m is an accepting computation, or every computation of $\Pi(n)$ with input m is a rejecting computation.

Proposition 1. *Let* $\mathcal{F}$ *be a class of recognizer P systems. Let* X, Y *be problems such that* X *is reducible to* Y *in polynomial time. If* $Y \in \mathbf{PMC}_{\mathcal{F}}$*, then* $X \in \mathbf{PMC}_{\mathcal{F}}$.

For a detailed proof, see [9].

5 Solving CADP by Recognizer P Systems

We will address the resolution of this problem via a brute force algorithm, in the framework of recognizer P systems with active membranes using 2-division, and without cooperation nor priority among rules. Our strategy will consist in the following phases:

- *Generation stage:*
  1. At the beginning there will be only one internal membrane which will represent the set $A = S$.
  2. `For each subset` $B \in F$ `do:`
     `if` $B \subseteq A$ `then`
     `for each` $e \in B$ `do:`
     `Using membrane division generate one membrane representing the subset` $A - \{ e \}$
     `end if`
- *Calculation stage:* In this stage the system calculates the cardinality of the subset associated with each membrane.

– *Checking stage:* Here the system checks whether or not the cardinality of each generated subset exceeds the goal k.
– *Output stage:* According to the previous stage, one object YES or one object NO is sent out to the environment.

Now we construct a family of P systems with active membranes using 2-division solving the *Common Algorithmic Decision Problem.*
Let us consider a polynomial bijection, $\langle\ \rangle$, between $\mathbb{N}^3$ and $\mathbb{N}$ (e.g., $\langle x,\, y,\, z\,\rangle = \langle\,\langle x,\, y\,\rangle,\, z\,\rangle$) induced by the pair function $\langle x, y\rangle = (x+y)\cdot(x+y+1)/2 + x$).

The family of P systems with input considered here is

$$\mathbf{\Pi} = \{\ (\Pi(\langle n,m,k\rangle), \Sigma(n,m,k), i(n,m,k))\ :\ (n,m,k) \in \mathbb{N}^3\ \}$$

For each $(n,m,k) \in \mathbb{N}^3$, we have $\Sigma(n,m,k) = \{\ s_{ij}\ :\ 1 \le i \le m,\ 1 \le j \le n\}$, $i(n,m,k) = 2$, and $\Pi(\langle n,m,k\rangle) = (\Gamma(n,m,k), \{1,2\}, \mu, \mathcal{M}_1, \mathcal{M}_2, R)$ is defined as follows:

– Working alphabet:

$$\begin{aligned}\Gamma(n,m,k) = \Sigma(n,m,k) \cup\ & \{a_i\ :\ 1 \le i \le m\ \} \cup \{c_i\ :\ 0 \le i \le 2n+1\ \}\\ & \cup \{ch_i\ :\ 0 \le i \le 2k+1\ \} \cup \{f_j\ :\ 1 \le j \le n+1\ \}\\ & \cup \{e_{i,j,l}\ :\ 1 \le i \le m, 1 \le j \le n, -1 \le l \le j+1\ \}\\ & \cup \{g_j\ :\ 0 \le j \le nm+m+1\ \}\\ & \cup \{z,\, s_+,\, s_-,\, S_+,\, S_-,\, S,\, o,\, \tilde{O},\, O,\, p,\, t,\, neg,\, i_1,\, i_2\}\\ & \cup \{YES_0,\, YES_1,\, YES_2,\, YES,\, preNO,\, NO\ \}.\end{aligned}$$

– Membrane structure: $\mu = [\ [\ \ ]_2\ ]_1$ (we will say that every membrane with label 2 is an *internal membrane*).
– Initial multisets: $\mathcal{M}_1 = \emptyset$; $\mathcal{M}_2 = \{g_0, z^m, s_+^n, o^k\}$.
– The set of evolution rules, R, consists of the following rules:

(1) $[\ s_{1,\,j}\ \to\ f_j\ ]_2^0$, for $1 \le j \le n$,
$[\ s_{i,\,j}\ \to\ e_{i,\,j,\,j}\ ]_2^0$, for $2 \le i \le m,\ 1 \le j \le n$.
The objects $s_{i,\,j}$ encode in the initial configuration the *forbidden sets*. The presence of an object $s_{i,\,j}$ indicates that $s_j \in B_i$. The objects of type f represent the elements of the *forbidden set* that is being analized and the objects e represent the rest of the *forbidden sets*.

(2) $[\ f_1\ ]_2^0 \ \to\ [\ \sharp\ ]_2^0\ [\ s_-\ ]_2^+$.
The goal of these rules is to generate membranes for subsets A of S such that $\forall\, B \in F\,(\,B \not\subseteq A\,)$. The system, in order to ensure the condition $B \not\subseteq A$, eliminates from A one element of the *forbidden set* B.

(3) $[\ f_{j'}\ \to\ f_{j'-1}\ ]_2^0$, for $2 \le j' \le n+1$,
$[\ e_{i,\,j,\,l}\ \to\ e_{i,\,j,\,l-1}\ ]_2^0$, for $2 \le i \le m,\ 1 \le j \le n,\ 0 \le l \le j+1$.
During the computation for a *forbidden subset*, B, the above rules perform a rotation of the subscript of the objects f and of the third subscript of the objects e. These subscripts represent the order in which the elements are considered to be eliminated from the subset A associated with the membrane in order to ensure the condition $B \not\subseteq A$.

(4) $[\ f_{j'}\ \to\ \sharp\ ]_2^+$, for $1 \le j' \le n+1$,
$[\ e_{i,\,j,\,0}\ \to a_{i-1}\ ]_2^+$, for $2 \le i \le m,\ 1 \le j \le n$.

When the polarization of an internal membrane is positive during the *generation stage* the associated subset A fulfills the condition $B \nsubseteq A$, where B is the *forbidden set* that is being analized. In this situation the elements of the current *forbidden set* are *erased* by these rules. Moreover, if the element removed from A is a member of the *forbidden set* B_i, then the object a_{i-1} appears in the membrane to certify that the associated subset also fulfills the condition $B_i \nsubseteq A$.

(5) $[\, e_{i,\,j,\,-1} \;\rightarrow\; e_{i-1,\,j,\,j+1} \,]_2^+$, for $3 \leq i \leq m$, $1 \leq j \leq n$,
$[\, e_{i,\,j,\,l} \;\rightarrow\; e_{i-1,\,j,\,j+1} \,]_2^+$, for $3 \leq i \leq m$, $1 \leq j \leq n$, $1 \leq l \leq j+1$,
$[\, e_{2,\,j,\,l} \;\rightarrow\; f_{j+1} \,]_2^+$, for $1 \leq j \leq n$, $1 \leq l \leq j+1$,
$[\, e_{2,\,j,\,-1} \;\rightarrow\; f_{j+1} \,]_2^+$, for $1 \leq j \leq n$,
$[\, a_{i'} \;\rightarrow\; a_{i'-1} \,]_2^+$, for $2 \leq i' \leq m$.

In order to continue the computation for the next *forbidden subset* B, these rules perform a rotation of the subscripts of the objects e and a. Note that the subscript representing the position of the element to be analized is set one position ahead in order to allow the system to check whether the current associated subset A satisfies the condition $B \nsubseteq A$.

(6) $[\, a_1 \,]_2^0 \;\rightarrow\; \sharp\, [\;\,]_2^+$; $[\, a_1 \;\rightarrow\; \sharp\,]_2^+$.

The presence of object a_1 in a neutrally charged internal membrane means that the *forbidden set* which is going to be analized already satisfies the condition $B \nsubseteq A$; consequently this object changes the polarization of the membrane to positive in order to skip the computation for this *forbidden subset*.

(7) $[\, z \,]_2^+ \;\rightarrow\; \sharp\, [\;\,]_2^0$.

The object z sets the polarization of the internal membranes to neutral once the generation stage for one forbidden set has taken place.

(8) $[\, g_j \;\rightarrow\; g_{j+1} \,]_2^0$, $[\, g_j \;\rightarrow\; g_{j+1} \,]_2^+$, for $0 \leq j \leq nm+m$,
$[\, g_{nm+m+1} \;\rightarrow\; neg,\, c_0 \,]_2^0$.

The objects g are counters used in the *generation stage*.

(9) $[\, neg \,]_2^0 \;\rightarrow\; \sharp\, [\;\,]_2^-$, $[\, z \,]_2^- \;\rightarrow\; \sharp$.

The multiplicity of the object z represents the number of *forbidden sets* that do not satisfy the condition $B \nsubseteq A$. So, if there is an object z in an internal membrane when the *generation stage* is over, then the membrane is dissolved.

(10) $[\, s_+ \;\rightarrow\; S_+ \,]_2^-$, $[\, s_- \;\rightarrow\; S_- \,]_2^-$, $[\, o \;\rightarrow\; \tilde{O} \,]_2^-$.

The multiplicity of the object s_+ represents the cardinality of S, the multiplicity of the object s_- represents the number of removed elements from S and the multiplicity of the object o represents the constant k. At the begining of the *calculation stage* the objects s_+, s_-, and o are renamed to S_+, S_-, and $\tilde{O}$ in order to avoid the interference with the previous stage.

(11) $[\, S_- \,]_2^- \;\rightarrow\; \sharp\, [\;\,]_2^+$, $[\, S_+ \,]_2^+ \;\rightarrow\; \sharp [\;\,]_2^-$.

These rules are used to calculate the cardinality of the subsets associated with the internal membranes.

(12) $[\, c_i \;\rightarrow\; c_{i+1} \,]_2^-$, $[\, c_i \;\rightarrow\; c_{i+1} \,]_2^+$, for $0 \leq i \leq 2n$,
$[c_{2n+1} \;\rightarrow\; t,\, ch_0]_2^-$.

The objects c are counters used in the *calculation stage*.

(13) $[\, t \,]_2^- \;\rightarrow\; \sharp\, [\;\,]_2^0$, $[\, S_+ \;\rightarrow\; S \,]_2^0$, $[\, \tilde{O} \;\rightarrow\; O \,]_2^0$.

The object t will change the polarization of the internal membranes to neutral starting the *checking stage*. In this stage the objects S_+ and $\tilde{O}$ are renamed to S and O, in order to avoid the interference with the previous stages.

(14) $[\, S\,]_2^0 \;\rightarrow\; \sharp\,[\;\;]_2^+,\, [\, O\,]_2^+ \;\rightarrow\; \sharp\,[\;\;]_2^0$

These rules are used to compare the multiplicity of the objects S and O.

(15) $[\, ch_i \;\rightarrow\; ch_{i+1}\,]_2^0,\, [\, ch_i \;\rightarrow\; ch_{i+1}\,]_2^+, \quad 0 \leq i \leq 2k.$

The objects ch are counters in the *checking stage*.

(16) $[\, ch_{2k+1}\,]_2^+ \;\rightarrow\; YES_0\,[\;\;]_2^-,\, [\, ch_{2k+1} \;\rightarrow\; p,\, i_1\,]_2^0.$

The *checking stage* finishes when the object ch_{2k+1} appears in the internal membranes. These rules are used to send the object YES_0 to the skin, if the charge is positive, and to check whether the answer must be NO.

(17) $[\, p\,]_2^0 \;\rightarrow\; \sharp\,[\;\;]_2^+,\, [\, i_1 \;\rightarrow\; i_2\,]_2^+,$
$[\, i_2\,]_2^+ \;\rightarrow\; YES\,[\;\;]_2^-,\, [\, i_2\,]_2^0 \;\rightarrow\; preNO\,[\;\;]_2^-.$

These rules decide if there are any objects O in the internal membranes when the *checking stage* is over, in order to send out the right answer.

(18) $[\, YES_i \;\rightarrow\; YES_{i+1}\,]_1^0$, for $0 \leq i \leq 1,$
$[\, YES_2 \;\rightarrow\; YES\,]_1^0,\, [\, preNO \;\rightarrow\; NO\,]_1^0.$

These rules are used to sinchronize the *output stage.*

(19) $[\, YES\,]_1^0 \;\rightarrow\; YES\,[\;\;]_1^+,\, [\, NO\,]_1^0 \;\rightarrow\; NO\,[\;\;]_1^-.$

These rules send out the answer to the environment.

6 An Overview of the Computation

First of all we define a polynomial encoding for the $CADP$ in $\mathbf{\Pi}$ in order to study its computational complexity. Let $h(u) = \langle\, n,\, m,\, k\,\rangle$ and $g(u) = \{\, s_{i,\,j} : s_j \in B_i\,\}$, for a given $CADP$–instance $u = (\{\, s_1,\, \ldots,\, s_n\,\}, (B_1,\, \ldots,\, B_m),\, k)$. Next we informally describe how the system $\Pi(h(u))$ with input $g(u)$ works.

In the first step of the computation, according to the rules in (1), the objects s evolve to the objects f and e. The objects f represent the elements of the *forbidden set* that is being analized and the objects e represent the others.

The generation stage takes place following the rules from group (2) - (8). The systems generates subsets of S with the greatest possible cardinalities and associates them with internal membranes. Let us describe the evolution of the *subsets associated with internal membranes* during the *generation stage*.

The *subset associated* with the initial internal membrane is $A = S$.

When the object f_1 appears in a neutrally charged internal membrane, during the *generation stage* for a *forbidden set* B, using the rule in (2) the system produces two new membranes: one (positively charged) where the analized element is removed, and another one (neutrally charged) where an element of B (different from f_1) is removed in order to achieve the condition $B \nsubseteq A$. These two new membranes behave in a different way.

On the one hand in order to study the next element, in the neutrally charged membrane the rules in (3) perform a rotation of the subscript of the objects f and of the third subscript of the objects e, which represent the order in which the elements are considered.

On the other hand, in the positively charged membrane an object s_- appears, indicating that one element has been removed from the associated subset A. The system moves on to analize the remaining *forbidden sets* rotating the first subscripts of the objects e according to the rules in (4) and (5), and *erasing* the objects f. Note that the third subscript of the objects e and the subscript of the object f are set one position ahead in order to check whether the condition imposed by the *forbidden set* is satisfied. The computation corresponding to a *forbidden set* finishes when an object z changes the polarization from positive to neutral following the rule in (7).

In the first step of the computation for a *forbidden set* the position of the elements in which they will be studied are one position ahead and so the system has a step to check whether there exists an object a_1 which means that the associated subset already fulfills the condition $B \nsubseteq A$. This is done applying the rules in (6). The objects a appear by applying the second rule in (4) when the system removes an element belonging to several *forbidden sets*.

The *generation stage* ends when the object g_{nm+m+1} appears. Between the *generation stage* and the *calculation stage* there is a gap of two steps of transition. In the first step, according to the last rule in (8), the object g_{nm+m+1} evolves to the object c_0 (a counter for the *calculation stage*) and the object *neg*. This object changes the polarization of the internal membranes to negative using the first rule in (9). In the second step, one dissolves the membranes whose associated subsets A have the property that there exists $B \in F$ such that $B \subseteq A$. The number of sets in F verifying $B \subseteq A$ is represented by the multiplicity of the object z. So, at the end of the *generation stage*, when the polarization is negative, if there is an object z, then the membrane is dissolved by the second rule in (9). Moreover, in this step the objects s_+, s_-, and o are renamed to S_+, S_-, and $\tilde{O}$ by the rules in (10), in order to avoid interference with the previous stage.

The multiplicity of the object S_+ encodes the cardinality of the set S and the multiplicity of the object S_- encodes the number of elements that have been removed from S to construct the final associated subset A. Thus, in order to calculate the cardinality of A the system applies the rules in (11), which implement the subtraction $multiplicity(S_+) - multiplicity(S_-)$ in each internal membrane.

The *calculation stage* ends when the object c_{2n+1} evolves to the object ch_0 (a counter in the *checking stage*) and the object t. By using the rule in (13), t changes the polarization of the internal membranes from negative to neutral.

In the transition stage from the *calculation stage* to the *checking stage* the objects S_+ and $\tilde{O}$ are renamed to S and O using the rules in (13) in order to avoid interference with the previous stages.

In the *checking stage*, by using the rules in (14), the system decides in each internal membrane if the multiplicity of the object S, encoding the cardinality of the *associated subset*, is greater than or equal to the multiplicity of the object O, encoding the constant k.

The *checking stage* ends when the object ch_{2k+1} appears in the internal membrane and then the *output stage* starts. If the object ch_{2k+1} appears when

the membrane is positively charged, then the number of objects S exceeded the number of object O and so, by using the first rule in (16), the object YES_0 is sent to the skin region. This object has to evolve to YES_1, YES_2 and, finally, to YES in order to synchronize the *output stage*. On the other hand, if the object ch_{2k+1} appears in a neutrally charged membrane, then the number of objects S was less than or equal to the number of objects O. In this situation the object ch_{2k+1}, following the second rule in (16), evolves to i_1 and p. This object changes the polarization of the internal membranes to positive in order to allow any remaining objects O to set it again to neutral according to the rules in (14). While the membrane is positively charged the object i_1 evolves to i_2. If i_2 appears in a positively charged internal membrane, then there were no objects O, therefore the multiplicity of the object S was equal to the multiplicity of the object O, and so the object YES is sent to the skin region according to the rules in (17). On the other hand, if the object i_2 appears in a neutrally charged membrane, then there were objects O and so the object $preNO$ is sent to the skin region.

In the last step of the computation the rules in (19) send out the answer to the environment. Note that the occurrence of the objects NO is delayed one step, by the rule $[\ preNO \rightarrow NO]_1^0$, in order to allow the system to send out the object YES, if any.

7 Required Resources

The presented family of recognizer P systems solving the *Common Algorithmic Decision Problem* is polynomially uniform by Turing machines. Note that the definition of the family is done in a recursive manner from a given instance, in particular, from the constants n, m, and k. Futhermore, the resources required to build an element of the family are the following:

- Size of the alphabet: $n^2m + 4nm + 2n + 3m + 2k + 24 \in O((max\{n, m, k\})^3)$.
- Number of membranes: $2 \in \Theta(1)$.
- $|\mathcal{M}_1| + |\mathcal{M}_2| = n + m + k + 1 \in O(n + m + k)$.
- Sum of the rules' lengths: $12n^2m + 12n^2 + 49nm + 71n + 25m + 30k + 242 \in O((max\{n, m, k\})^3)$.

The instance $u = (\{ s_1, \ldots, s_n \}, (\{ s_1^1, \ldots, s_{r_1}^1 \}, \ldots, \{ s_1^m, \ldots, s_{r_m}^m \}), k)$ is introduced in the initial configuration through an input multiset; that is, it is encoded in an *unary representation* and, thus, we have that $|u| \in O(n + m + k)$.

The number of steps in each stage are the following:

1. *Generation stage:* $nm + m + 1$ steps.
2. *Transition to the calculation stage:* 2 steps.
3. *Calculation stage:* $2n + 1$ steps.
4. *Transition to the checking stage:* 2 steps.
5. *Checking stage:* $2k + 2$ steps.
6. *Output stage:* 6 steps.

So, the overall number of steps is $nm + 2n + m + 2k + 14 \in O(max\{n, m, k\}^2)$.

From these discussions we deduce the following results:

Theorem 7. $CADP \in \mathbf{PMC}_{\mathcal{AM}}$.

Corollary 1. $\mathbf{NP} \subseteq \mathbf{PMC}_{\mathcal{AM}}$, *and* $\mathbf{NP} \cup \mathbf{co} - \mathbf{NP} \subseteq \mathbf{PMC}_{\mathcal{AM}}$.

8 Conclusions

Many **NP**-complete problems can be viewed as special cases of an optimization problem called Common Algoritmic Problem. In this work the importance of this problem is emphasized by the presentation of six relevant **NP**-complete problems that are "subproblems" of it (or of its corresponding decision version). Furthermore, a solution to the Common Algoritmic Decision Problem by a family of recognizer P systems with active membranes is presented.

The study and design of solutions to *locally universal* problems as CAP and CADP within the framework of unconventional computing models like P systems seems very interesting because these solutions may give, in some sense, *patterns* that can be used for attacking the solvability of many **NP**-complete problems.

References

1. Gutiérrez–Naranjo, M.A.; Pérez–Jiménez, M.J.; Riscos–Núñez, A. Towards a programming language in cellular computing. In: Gh. Păun; A. Riscos–Núñez, A. Romero–Jiménez; F. Sancho–Caparrini (eds.) *Proceedings of the Second Brainstorming Week on Membrane Computing*, Report RGNC 01/04, University of Seville, Spain, 2004, 247–257.
2. Head, T.; Yamamura, M.; Gal, S. Aqueous computing: writing on molecules. *Proceedings of the Congress on Evolutionary Computation 1999*, IEEE Service Center, Piscataway, NJ, 1999, 1006–1010.
3. Păun, Gh.: *Membrane Computing. An Introduction*, Springer-Verlag, 2002
4. Păun, Gh.: Computing with membranes. *Journal of Computer and Systems Sciences*, 61(1), 2000, 108–143.
5. Pérez–Jiménez, M.J.; Romero–Jiménez, A.; Sancho–Caparrini, F.: *Teoría de la Complejidad en modelos de computacion celular con membranas*, Ed. Kronos, 2002.
6. Pérez-Jiménez, M.J.; Riscos-Núñez, A. Solving the Subset-Sum problem by active membranes, submitted.
7. Pérez–Jiménez, M.J.; Riscos–Núñez, A. A linear-time solution for the Knapsack problem using active membranes. *Lecture Notes in Computer Science*, 2933 (2004) 140–152.
8. Pérez–Jiménez, M.J.; Romero–Campero, F.J. A CLIPS simulator for recognizer P systems with active membranes. In: Gh. Păun; A. Riscos–Núñez; A. Romero–Jiménez; F. Sancho–Caparrioni (eds.) *Proceedings of the Second Brainstorming Week on Membrane Computing*, Report RGNC 01/04, University of Seville, Spain, 2004, 387–413.
9. Pérez–Jiménez, M.J.; Romero–Jiménez, A.; Sancho–Caparrini, F. A polynomial complexity class in P systems using membrane division. In: E. Csuhaj-Varjú; C. Kintala; D. Wotschke; Gy. Vaszil (eds.) *Proceedings of the Fifth International Workshop on Descriptional Complexity of Formal Systems*, 2003, 284–294.